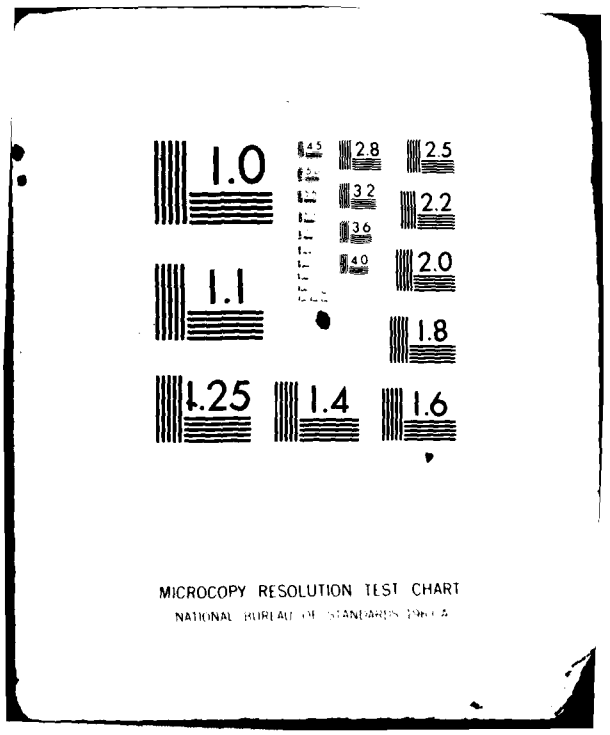
AD-A115 981 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 9/2
QUERY PROCESSING IN DISTRIBUTED HETEROGENEOUS DATABASES,(U)
DEC 81 K HUANG, W B DAVENPORT N00014-77-C-0532
UNCLASSIFIED LIDS-P-1212 NL

END
DATE FILMED
7-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

(12)

December 1981 LIDS-P-1212

Query Processing in Distributed Heterogeneous Databases**

by

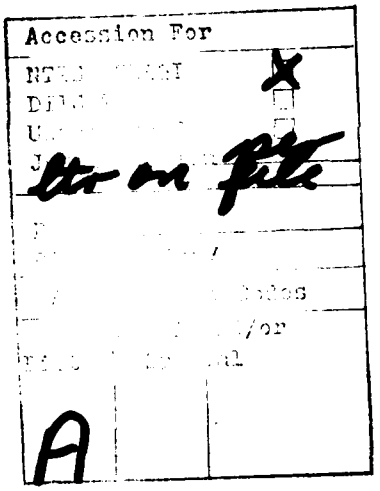
Kuan-Tsae Huang*
Wilbur B. Davenport, Jr.

* The authors are with the Dept. of Electrical Engineering and Computer Science and the Laboratory for Information and Decision Systems at Mass. Institute of Technology, Cambridge. Mass. 02139.

** This research was conducted at the MIT Laboratory for Information and Decision Systems with support provided by the Office of Naval Research under contract ONR/N00014-77-C-0532.

Abstract

In order to bring distributed heterogeneous DBMSs which are implemented with relational, CODASYL or hierachical models together to share the information resources and build new kinds of applications, we propose to use relational model as global conceptual model and ALPHA as global nonprocedual high-level query language. We described the distributed query processing scheme in this environment. A schema translation and query translation for CODASYL system case are discussed. Relational and Hierachical systems can be treated similarily.



1. Introduction

Database management systems are amongst the most important and successful software developments in this decade. They have already had a significant inpact in the field of data processing and information retrieval. The existing commercial systems are almost exclusively based on one of the three data models: relational, CODASYL and hierachical. Many organizations have developed independently their own databases on their own computers and database management systems based on one of the three data models to support the planning and decision making in operations. Each DBMS has its own intended schema. access control. degree of efficiency, security classification and operational requirements, etc. Often, different databases may contain data relevant to the problem although their structure and representation could be different. If we can bring together all these databases in several locations in order to integrate information resources and build new kinds of applications to help operations. it will be beneficial.

Prior work on this problem have used two approaches. One is to convert and to migrate the entire database from one type of system to the other type of system. [SHL 75] and [SU 76] are works in this direction. The other approach is to maintain the original database and provide an effective information exchange among the different systems without incurring mass data migration. A global data model is used to provide users with a common schema. Each data base can both operate in its own local

mode and participate in the distributed system. [AD77], [CP 80] ans [SBD 81] are in this direction. Our approach [HD 81] can be classified into the second category. The main differences of our approach from others are that we take advantage of the relational data model and use a very high-level non-procedural language as a common language for a user interface.

Although many logical data models have been proposed which model the real world in terms of the interested objects and the interrelation between them. it is clear that there is no mental model which is good for all users. We choose a relational data model as global data model in order to provide a central view to the users bases on the following reasons:

1. The relational data model shields the user from data formats. access methods and the complexity of storage structures.

2. It supports a high-level non-procedural query language.

3. The storage and data structures are very simple. all data is represented in the form of records.

4. Access paths do not have to be predefined. A number of power operators are supported in relational model. e.g., select, project. join. etc. for data retrieval.

5. Because of the decline of hardware costs and the rise of manpower costs. a high-level nonprocedural manipulation language is necessary in orderto minimize the user workload.

6. The relational model provides a simpler and powerful interface to the data.

7. The relational model has a fast response to ad hoc

queries which are considered to be high-percentage of queries.

8. The advance in associative storage devices offers the potential of greatly improving the efficiency and therefore the performance of relational system.

Our basic underlying assumptions are: 1. It is possible to exchange information amongst the various system and they are willing to maintain information. 2. Each DBMS is condsidered to be able to execute a given local transaction. 3. There exists a communication network which connects the various DBMSs. 4. the access to a local DBMS is not affected by the operation of the data communication system which should transparent to the local user. 5. The communication cost is the dominant factor and hence local processing is essentially free.

2. Query Processing in Heterogeneous Environment

Based on the architecture of the database communication system (DCS) we proposed in [HD81] . we adopt the relational model as the global conceptual model. It is necessary to provide a relational schema for each database. For those databases in which the underlying data models are not relational models. schema translators will be required to do the translation jobs. A specific schema translator is needed for a specific data model to translate the underlying schema to a relational logical schema. The integration schema consists of information about integrity constraints. data incompatibility and data redundancy. This integration schema can be viewed as a small database. For the

query which is against this local relational schema. it is also necessary to provide translation rules to translate the relational operations into data manipulation language statements of the underlying data model. The users will see the system as a distributed relational database system.

In our approach, each database system is presented to a user with a global relational schema. A query for data access or update is specified in terms of a relational calculus-like qualification over relations with a target list. Codd's data sublanguage ALPHA (DSL ALPHA) [CODD 72] is one of the calculus-based data sublanguages. It consists simply of the relational calculus in a syntatic form which more closely resembles that of a programming language. In practice, the syntax would have to be compatible with that of the host language, whatever that was. For our purpose. we shall use the syntax of ALPHA which is expressed in [CODD 70]. An example of a query expressed in ALPHA is:

EXAMPLE:

Let the query be "Get SNO values for suppliers who supply a LONDON or PARIS project with a red part". The corresoponding ALPHA statement of this query will be:

```
RANGE P PX
RANGE J JX
GET W (SPJ.SNO) :   PX  JX(PX.COLOR='RED'
                    ∧(JX.CITY='LONDON'      ∨      JX.CITY='PARIS')
                    ∧SPJ.PNO=PX.PNO∧SPJ.JNO=JX.JNO)
```

The list of attributes within parenthesis of W is the target list which specifies the attributes to retrieve. The predicate calculus following ":" is the qualification

The query optimizer in the query processing unit of the system will transform the query into an optimal sequence of relational algebra operations. In order to execute the operations against the local DBMS, we have to translate a relation algebra operation against this DBMS to a program of DML statements of the target system. The query translator plays the role of this process.

We assume in this paper that the transmission mode of any system be all-records-at-a-time, the form of data after being retrieved from local DBMS and temporarily stored in user working area (UWA) of each local DBMS are in relational able-like form and each DCS has the ability to execute relational algebra operations. These assumptions will enable easier data transmission when it is required. In the next section. we address the problem of designing a schema translator and a query translator of CODASYL data model. Relational and hierachical systems can be treated similarily. We focus on those changes that must be made because of the different in the level of procedurality of the relational algebra operators and data manipulation language of target data models.

3. Network model case

A great deal of attention has been focused on the network approach since the publication in April 1971 of the CODASYL DBTG final report [CODA 71]. A number of commercially available systems have used one or more versions of the specifications as the implementation base. While those commercial implementation may show slight differences, their underlying concepts are based on the same CODASYL/DBTG data model.

3.1 Schema translation

The schema is the logical description of the data base. A schema description in th DBTG DDL includes four types of declarations: The schema name description. Record type declarations, Set declarations and Area declarations.

There are two kinds of record types in the CODASYL/DBTG model: a description record type and a connection record type. A description record type in the DBTG data model has a record ID, and one or several attributes describing properties of the record. It is very similar to a relation in the relational data model with the record ID as the key attribute. Therefore, a description record type can be translated to a relational schema directly. A connection record type is introduced when n types of entity (represented by n description record types) are to be connected. N set types also introduced. Each of the n "entity" record types is made the owner of one of the set types, and the connection record type is made the member of them. Each connection record occurrence is made a member of exactly one

occurrence of each of the n types of set and thus represents the connection between the corresponding n entities. For this record type we define a relation schema R with attributes consisting of the keys of the owners of the n sets in which this record is a member and the data items of this record. The key of this relation is the set of keys of the owners of the n sets.

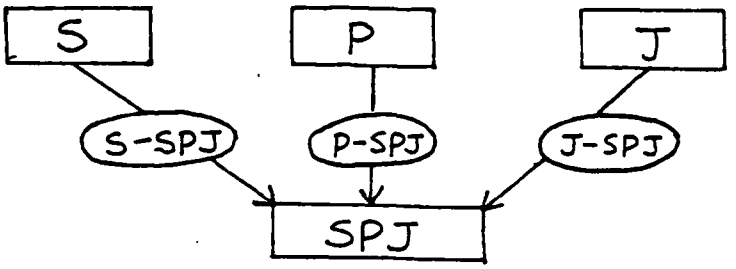
A set type is defined in the schema to have a certain type of record as its owner and some other type of record as its member. Each occurence of a set type consists of precisely one occurrence of its owner together with zero or more occurences of its member. For each set type, we do not correspondingly define a relational schema. We create a table in each node to record all the sets which can be thought of as an access path relation. This table contains three attributes, { set, owner, member }. It is used in the query translation process from a relational operator to CODASYL DML statements in order to identify the access path. The table can be thought of as a new relation which is stored in the local database and only used for query translation. This table provides information of record access paths. It isn't joined to the global schema to be presented to the user.

An area is a storage space of a DBTG database. For each type of record the schema specifies the area into which occurrences of the record are to be placed when they are entered into the database. This area type is correspondingly mapped to the horizontal or vertical partition of a relational database which may require to create a new relational schema. In this paper, we

shall assume that all occurrences of a given type of record are to go into a single area.

EXAMPLE [DATE 77]

Assume the supplier-part-project database S is stored in a CODASYL version DBMS. The schema of this database is defined as follows.



After applying the rules of schema translation. we define the following relational schema:

S=(SNO. SNAME. STATUS, CITY)

P=(PNO. PNAME. COLOR, WEIGHT)

J=(JNO. JNAME. CITY)

SPJ=(SNO. PNO. JNO. QTT).

We create an access path relation as follows:

set	owner	member
S-SPJ	S	SPJ
J-SPJ	J	SPJ
P-SPJ	P	SPJ
S	SYSTEM	S
P	SYSTEM	P
J	SYSTEM	J

3.2 Query Translation

A query in CODASYL model is a sequence of DML statements which are embedded within a program as a syntactic extension of the host language. The details of CODASYL DML are described in [DAT 77]. In the distributed DBMS environment. we assume each system has a SEND command which can send a file or part of UWA from one system to another system. To update a database, or to create a new record, it is required to retrieve the data occurrence to be updated or to create a new data occurrence in UWA. After updating the data occurrence in UWA. a MODIFY or STORE statement is then applied. Therefore, we only need to consider retrieve operations. We first study the translation procedures for two unary algebra operators: project and select.

We assume $R=\{A_1.A_2.....A_n\}$, $X=\{A_1.....A_K\}$. PROJECT. denoted by $\Pi_X(R)$. is to retrieve the attributes in X of each record in R. The algorithm $PROJECT_N(R,X)$. as shown below is to translate $\Pi_X(R)$ into a DML program.

```
PROJECT_N(R.X):
   IF ( Π{set}(t)='R-set' and Π{owner}(t)='system')
      /* description record */
   THEN      NXT:   FIND next R within R-set;
                    IF end of set GOTO quit;
                    GET A1 IN R. ...., AK IN R;
                    GOTO NXT.
   ELSE IF ( Π{set}(t)='S-R' and Π{owner}(t)='S')
```

```
        /* connection record */
    THEN     NXT: FIND next R within S-R;
                  IF end of set GOTO quit;
                  GET A1 in R,.... AK in R.
                  GOTO NXT.
```

Let $Y=\{A_1, A_2, \ldots, A_\ell\} \subseteq R$. SELECT. denoted by $\sigma_{\{X A_i = c_i'\}}(R)$, is to select tuples in R such that $R.A_1='c_1'$..... $R.A_K='c_K'$. The algorithm below is the translation algorithm.

```
SELECT(R,Y,c1,c2,...,cK):
   IF (R is a connection record)
   THEN if (some Ai is and attribute of S) & (S-R is a set)
      THEN BEGIN
          MOVE 'c1' to A1 in S;
          FIND any S;
          IF S-R empty GOTO quit;
          MOVE 'c2' to A2 in R;
                 :
                 :
          MOVE 'cK' to AK in R;
          FIND any R within S-R;
          IF end of set GOTO quit;
          GET R;
          END;
     ELSE BEGIN
          /* there exist some S s.t. S-R is a set */
         NXT: FIND next S within S-set;
```

```
        IF end of set GOTO quit;
        GET A  in S;
        MOVE  'c1' to A1 in R;
              :
              :
        MOVE  'cK' to AK in R;
        FIND any R within S-R;
        GET A1 in R. ... , Al in R;
        IF end of set GOTO NXT;
    END;
```

When the two operands of the operator are in different systems. we always retrieve the first operand and store in user working area as a relation and then send it to the other system to produce the final result. We consider a binary operator. join, as an example. Let R_1 and R_2 be two relational schemas at different systems S_1 & S_2respectively and $X=R_1 \cap R_2$ be the set of attributes in R_1 and R_2. Without lost of generality. we assume $X=\{A_1,A_2,\ldots,A_K\}$. Let T_1 and T_2 be the temporary relations. The distributed join of R_1 and R_2 over X is a distributed query operator which executes the following sequence of operations: 1. retrieve R_2 from S_2 as T_2; 2. send T_2 from S_2 to S_1; 3. $R \underset{X}{|X|} T$ at site S_1. The query translation of this operator is as follow:

```
IF R1 and R2 are both relational DBMSs
   THEN execute it as relational operator;
   ELSE IF R1 is relational and R2 is CODASYL
```

THEN

$P_N(R_2,R_2)$ at S_2 as T ;

send T from S_2 to S_1 ;

execute $R_1 \underset{X}{|X|} R_2$ as relational operator;

ELSE IF R_1 is CODASYL

THEN

retrieve R_2 from S_2 as a elation T;

send T from S_2 to S_1 ;

For all $t \in T$

$SELECT_N(R_1, R_1-X, t_1, t_2, \ldots, t_K)$;

This operation will create a new relation T' with relation schema $R_1 \cup R_2$ at working area of system S_1.

Semijoin, Union, Intersection and Difference of two relations require two operands having the same set of attributes. They can execute similarily by using project and select to retrive data and put into relational form and then perform the binary operation. If both two operands are stored in relational systems then we just perform the relational operation. If one of the two operands R are stored in CODASYL system, then we use $PROJECT_N(R,R)$ to retrieve R to form a relation and perform relational operation afterward. If both two operands are stored in CODASYL systems, then we must consider two cases. If the results of the operation are temporarily stored in UWA for using by later operations, then we both use $PROJECT_N(R_1,R_1)$ and $PROJECT_N(R_2,R_2)$ to retrieve both two operands as two relations and send one relation from one site to the other to perform the operation;

or we can use PROJECT to retrieve one operand as a relation and send to the other site and then perform selections.

When the two operands of a binary operation are at the same site and are stored in a CODASYL system, then we can perform the operation as the same sequence we discussed above. However. we can do it in a more efficient way because it is not necessary to retrieve one operand as a relation first in the execution of this operation. We can use solely CODASYL DML statements to perform this operation and do it by record-at-a-time or set-at-a-time. In some cases, we even can do it much better by combining a sequence of relational opeations which reference data stored at the same CODASYL system and translating them to a sequence of CODASYL DML statements.

4. Conclusions

In this paper. we have presented detailed schema translation rules for CODASYL data model translation to relational data model and algorithms for translating a relational ALPHA query into CODASYL DML statements whose associated databases have themselves been translated. The translation algorithms are developed for each relational atomic operation. The translation procedures are based on the relational operations sequence provided by query optimizer with the objective of minimizing communication complexity and are done one relational algebra operation at a time, It also uses the information of access path relation. Some optimization for the local query processing can be made by

translating a subquery which is a continuing subsequence of relational operations which reference data at the same DBMS into a sequence of CODASYL DML statements rather than translating one operation at a time. Because our major concern is the communication cost. this local processing optimization are not considered in this paper.

REFERENCES

[AD 78]
Abida, M. and Portal, D. A cooperation system for heterogeneous data base management systems. Inform. Systems 3(3):209-215, 1978.

[CODA 71]
Data Base Task Goup of CODASYL programming language committee, Report, ACM Apr 1971.

[CODA 78]
CODASYL data description language committee, DDL Journal of Development 1978.

[CODD 70]
Codd, E. F. A relational model for large shared data bases. Comm. ACM 13(6):377-387, Jun 1970.

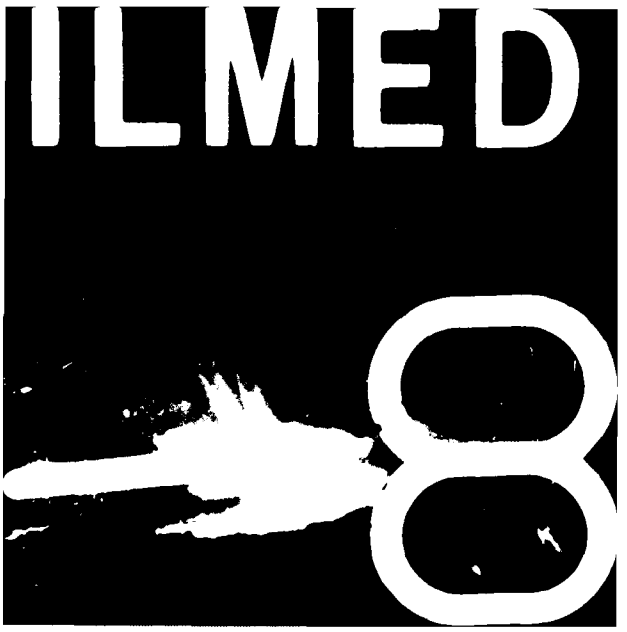
[CP 80]
Cardenas, A. and Pirahesh, M. H. Data base communication in a heterogeneous data base management system network. Inform. Systems 5(1):55-79, 1980.

[DAT 77]
Date, C. J. An Introduction to Database Systems, 2ed, Addison Wesley, Reading Mass. 1977.

[HD 81]
Huang,K.T. and Davenport, W.B. Issues in distributed database management systems. The 4th MIT/ONR Control, Command and Communication workshop. 1981.

[SHTGL 77]
Shu, N., Housel, B., Taylor, R. W., Ghosh, S. P., Lum, V. EXPRESS: a data extraction, processing and restructuring system. ACM TODS 2(2). Jun 1977.

[SLH 76]
Shu, N., Lum, V., Housel, B. An approach to data migration in computer networks. IBM Rep. RJ 1703 1976.


ILMED